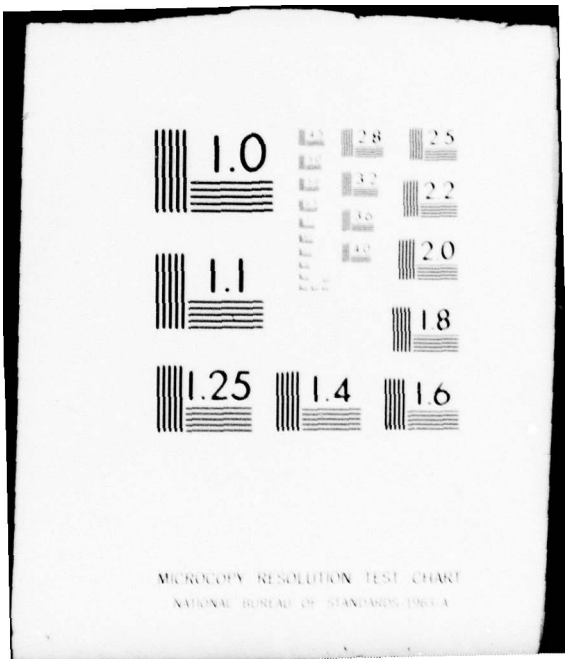
AD-A072 165 BOOZ-ALLEN AND HAMILTON INC BETHESDA MD COMMUNICATI--ETC F/G 17/2
INTEGRATED AUTODIN SYSTEM (IAS). GENERATION OF VIABLE FAR-TERM --ETC(U)
JUN 79 S SHRIER, P TORELLI DCA100-77-C-0057
UNCLASSIFIED SBIE-AD-E100 248 NL

| OF |
AD
A072165

END
DATE
FILMED
9-79
DDC


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

LEVEL III

AD-E100 248

FINAL REPORT

Integrated AUTODIN System (IAS)

GENERATION OF VIABLE FAR-TERM ARCHITECTURES FOR IAS

DCA100-77-C-0057

Task 1-78/1/5

June 1, 1979

Communications and Information Technology Div[illegible]

411 252

BOOZ ALLEN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Task 1-78/1/5 ✓	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Generation of Viable Far-Term Architectures for the Integrated AUTODIN System (IAS)	5. TYPE OF REPORT & PERIOD COVERED: Final Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): S. Shrier, P. Torelli	8. CONTRACT OR GRANT NUMBER(s): DCA100-77-C-0057 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Booz, Allen & Hamilton, Inc. 4330 East West Highway Bethesda, Maryland, 20014	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: N/A
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Engineering Center Computer Systems Architecture Branch, Code R810 1860 Wiehle Avenue, Reston, Virginia 22090	12. REPORT DATE: June 1, 1979 / 13. NUMBER OF PAGES: 37
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): N/A	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

A. Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Architecture Generation
IAS Architectures

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents an implementation of a methodology for the synthesis and ranking of logical architectures. The purpose of the methodology is to generate in a plausible manner, those logical architectures that lead to the identification and selection of candidate network architectures.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

EXECUTIVE SUMMARY

The planned development of an Integrated AUTODIN System (IAS) has been organized into three stages. These stages correspond to near-term (1978-1983), mid-term (1984-1988), and far-term (Post 1988) development. This report focuses on Far-Term IAS development. The Far-Term IAS is to embody worldwide implementation of emerging technologies of integrated voice and data and multiple access satellite broadcast capability.

The first step towards accomplishing Far-Term IAS development is the formulation of an architecture. An architecture is a definition of a system that describes the elements that comprise the system, how the system processes are distributed among the elements, and how these processes are combined to enable the system to provide its services to its users. The process of formulating a system architecture consists of determining the alternatives, developing a methodology to evaluate the alternatives, and evaluating the alternatives to choose an optimum one. In an earlier Booz, Allen report entitled "Structure of a Methodology for Generation of IAS Candidate Architectures", a four-phase methodology for determining Far-Term IAS architectural alternatives was presented. This report is a continuation of the earlier report. The purpose of this report is to present the mathematical approach for accomplishing the first phase of the methodology.

The four phases for determining Far-Term IAS architectural alternatives are the following:

- Synthesis and ranking of logical architectures - a logical architecture is a plausible sequence of functions
- Development of functional architectures - a functional architecture is a hierarchical stratification of a logical architecture
- Development of system architectures - a system architecture defines a set of functional elements in accordance with the stratification embodied in the functional architecture
- Development of network architectures - a network architecture is a refinement of a system architecture that specifies transmission systems, interconnection policy, protocols, and routing strategy.

The reason for developing this methodology is that the total set of feasible Far-Term IAS architectures is extremely large. Therefore, an initial screening process is required to arrive at a small set of candidate architectures which can then be subjected to a complete evaluation. The four-phase methodology derives candidate architectures in such a way that all possible architectures are at least implicitly considered.

This report develops the mathematical approach to synthesis and ranking of logical architectures. A logical architecture describes the system as a plausible sequence of functions, performed during the transfer of information through the network. These sequences are stated without reference to the system elements that might implement them. The only constraints on these sequences are constraints that are embodied in the definitions of the functions themselves. These constraints fall into the following two classes:

- The number of times that a function needs to be performed
- The order in which the functions are to be performed.

The set of functions and constraints can be expressed in the form of a regular grammar. The regular grammar is a mathematical construction that enables the set of logical architectures to be synthesized in a computationally efficient manner.

The ranking of logical architectures is based on the preferences of knowledgable observers. These preferences are aggregated as a set of weights. The weights are input to a dynamic programming algorithm that determines a set of optimum logical architectures. Since the optimization is performed over the space of all logical architectures, it follows that all logical architectures are at least implicitly considered in the ranking process.

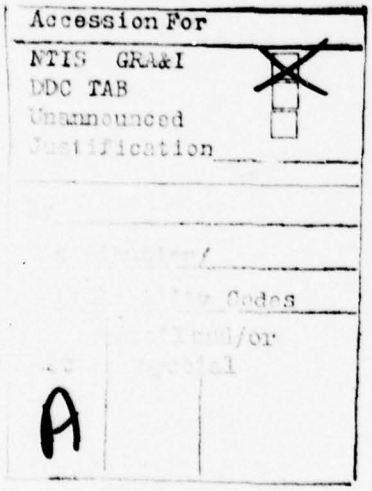


TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

GENERATION OF VIABLE FAR-TERM ARCHITECTURES FOR THE INTEGRATED AUTODIN SYSTEM

1. INTRODUCTION

a. Background. The Integrated Automatic Digital Network (AUTODIN) System (IAS) is a single, integrated, worldwide, high-speed, computer controlled, general purpose communications network that provides secure data, message, and record communications for the military services and Department of Defense (DoD) agencies. AUTODIN became operational in the early 1960's as a store-and-forward message switched network using leased commercial trunk lines.* This store-and-forward switched network is referred to as AUTODIN I. Although AUTODIN I has been continually enhanced over the years to reflect evolving communications needs, DoD decided that in order to meet the projected requirements of the 1980's and 1990's for computer and data communications, a packet switched network, AUTODIN II, should be developed and integrated with AUTODIN I (Reference 2). Furthermore, the General Accounting Office (GAO) has mandated that the growth and development of the AUTODIN System be accomplished in an evolutionary manner (Reference 3).

In order to accomplish this evolutionary transition, the development of IAS has been organized into three stages corresponding to the following time frames:

- Near-Term (1978-1983) - The near-term development consists of implementation of the AUTODIN II packet switched data network in CONUS and consolidation of this network with the existing AUTODIN I message switching network.

- Mid-Term (1984-1988) - The mid-term development consists of the expansion of the AUTODIN II packet switched network worldwide, closure of AUTODIN I switching centers, and introduction of standard terminals for all services and DoD Agencies

- Far-Term (Post 1988) - The far-term development consists of evolution toward the third generation Defense Communications System (DCS) which is to embody worldwide implementation of technologies for integrated voice and data and multiple access satellite broadcast capability.

---

* This report assumes familiarity with the basic notions and terminology of communication networks. These notions and terminology are explained in Reference 1.

The first step in the development of a communications network, given specified requirements, is the specification of a network architecture. Each of the three stages of the development of IAS requires such an architecture. Near-term and mid-term architectures are presented in References 4 and 5. Reference 6 initiated the development of a far-term architecture by presenting a four-phase methodology for generating and ranking alternative network architectures. This document describes the implementation of the first phase of this methodology and describes an illustrative example of this methodology.

The reason for developing a special methodology is that the set of feasible far-term architectures is extremely large because of the large number of choices of functional assignments, transmission media alternatives, and connectivity options available to the system architect. A consequence of the size of this architecture set is that it is nearly impossible to enumerate all feasible architectures. As a result, any definition and evaluation of a selected set of architectures may fail to consider an optimal alternative. Therefore, a methodology was sought to allow all possible architectures to be considered in an efficient and effective manner. The goal of the methodology is to arrive at a small set of candidate network architectures that is then subjected to a complete evaluation as to cost, benefits, and risks. A literature search failed to reveal an existing methodology that accomplishes this goal.

The whole process of defining a set of candidate network architectures is broken down into the following four phases:

- Phase 1: Synthesis and Ranking of Logical Architectures.

  A logical architecture is defined as a plausible sequence of functions. In this document, functions refer to those activities that must be performed in order to accomplish the transfer of information from one person or place to other persons or places (i.e., the steps in accomplishing the act of communications). The notion of a plausible set of functions is explained in Section 2.b. A logical architecture does not contain any reference to the network elements that would accomplish these functions. The notion of ranking logical architectures is discussed in Section 3.

- Phase 2: Development of Functional Architectures.

  A functional architecture is a hierarchical stratification of a logical architecture. Functions are grouped as a preliminary step to defining network levels. One or more groups of functions may be performed at a given level.

. Phase 3: Development of System Architectures.

A system architecture defines the communications network as a system of interconnecting nodes, and defines the nodes and the connections between the nodes in terms of the functions that they perform.

. Phase 4: Development of Network Architectures.

A network architecture is a further refinement of a system architecture that specifies transmission systems, interconnection policy, protocols, and routing strategies.

Following the completion of Phase 4, an evaluation is performed on a set of candidate network architectures to arrive at a preferred architecture. This evaluation includes a full analysis of the relative costs, benefits, and risks of the candidate alternatives.

As a consequence of Phase 1, a set of (or possibly a unique) candidate logical architectures is produced. The subsequent phases would then be carried out only for the candidate logical architectures. However, Phase 1 is accomplished in such a way that all possible logical architectures are at least implicitly considered. In this way the methodology assures that no alternatives are overlooked.

It should be noted that a given logical architecture may correspond to several network architectures. Therefore, the ranking of logical architectures is only the first step in arriving at a set of candidate network architectures.

b. Purpose. The purpose of this document is to present an implementation of a methodology for the synthesis and ranking of logical architectures. That is, a methodology for accomplishing Phase 1 of the four phase process indicated above. Approaches to accomplishing the subsequent phases of this process are indicated in Reference 6, however, implementation of methodologies for Phases 2, 3, and 4 have not yet been developed. The synthesis of logical architectures makes use of techniques of the theory of formal languages and abduction algorithms. The ranking process uses dynamic programming techniques.

c. Overview of Document. Section 2 presents a discussion of the four-phase process for defining candidate network architectures that was briefly indicated in Section 1.a. Section 3 is an explanation of the technical approach to the method for synthesis and ranking of logical architectures. Section 4 presents an application of the method. Appendix A presents the concepts from the theory of formal languages used in the methodology. Appendix B is a brief introduction to computer programs that support the methodology. Appendix C presents definitions of representative IAS functions.

2. OVERVIEW OF METHODOLOGY

The purpose of this section is to present an overview of the four phases of the methodology for defining a set of candidate architectures. Section 2.a sets the stage for this overview by discussing the functions that IAS is to be required to perform. These functions are the starting point for IAS architecture development. Sections 2.b through 2.e discuss each of the four phases in turn.

a. IAS Functions. Reference 6 contains a discussion of the required capabilities that IAS is to have and the services that are currently being considered for incorporation into IAS. The providing of these services requires that IAS be capable of performing a wide range of functions.* This set of functions is the starting point for architecture development.

The functions to be performed by IAS fall naturally into the following three categories:

- In-line functions - This category includes those functions that are usually in the direct sequence of events when a unit of information is transferred through a network from one subscriber to another.

---

* For the purpose of this report, functions are distinguished from services as follows:

- Services are entities that are specifically related to user requirements. These requirements are capable of being stated without use of communications terminology. Examples of services include: file retrieval, privacy service, facsimile, mailbox, and management and financial services.

- Functions are activities that a network performs in order to carry out the services. Examples of functions are address conversion, multiplexing, switching, data base maintenance, and traffic control.

Note that whereas a user is always aware of the services he is receiving, the functions that IAS performs in providing those services may be transparent to the user.

- Auxiliary functions - This category includes functions that are not directly encountered in a transaction flow through the network but which are required to support the in-line functions or to support certain network services.

- Management and control functions - This category includes those functions that are required to maintain and control the communications network.

As indicated in Reference 6, the capabilities and services offered by IAS translate into the set of functions to be performed by IAS. This translation process is described in Reference 6. Appendix C presents definitions of functions that might be performed by IAS.

b. Development of Logical Architectures. Logical architectures are defined as organizations or combinations of functions which are independent of network hierarchy or physical implementation. A logical architecture thus represents an arrangement of functions that are necessary for the basic process of communication to take place. For a given list of functions, the factors that distinguish one logical architecture from another are the number of times each function is to be performed and the order in which they are performed each time information is transmitted from one network subscriber to another.

For example, if there are five functions that the network must perform each time information is transferred, and if these five functions are denoted by $f_1,\ldots,f_5$, then some logical architectures may be represented as follows:

- $f_1\ f_3\ f_2\ f_1\ f_5\ f_4\ f_3$ (a)
- $f_1\ f_2\ f_4\ f_3\ f_5$ (b)
- $f_3\ f_2\ f_4\ f_2\ f_4\ f_4\ f_1\ f_5$ (c)

Thus, logical architectures may be represented as finite sequences of the f_i where each f_i must appear at least once in the sequence. The interpretation of these sequences is that in the process of communication, the functions are carried out in the order in which they appear in the sequence.

In general, logical architectures will be further restricted to those sequences of functions that satisfy a given set of constraints. These constraints pertain to the number of times and the order in which functions can be performed. For example, if the given set of constraints specified that each function is to be performed once and only once, then of the three sequences presented above, only (b) would be a logical architecture.

The purpose of the set of constraints is to assure that the logical architectures are plausible sequences of functions. The notion of plausibility arises from the fact that it may not make sense to perform the functions in certain orders. For example, Red/Black Conversion should always be performed before Black/Red Conversion, and Address Generation should always be performed before Address Conversion (See Appendix C for the definition of these functions). In addition, constraints may restrict the number of times functions can be performed and their placement in a sequence. For example, switching (See Appendix C) would never be the first or last function in a sequence.

Thus, logical architectures are plausible sequences of functions. The functions represented in a logical architecture are those functions that need to be carried out each time the communications process is to take place. All of these functions are in-line, implementation independent functions. A representative sample of functions that would be included in logical architectures is given in Table I. Given a set of functions and a set of constraints, the methodology in Section 3 can be applied to synthesize and rank the corresponding set of logical architectures. This ranking results in a set of logical architectures that are optimal, to the extent that optimization can be carried out at this high level.

TABLE I. REPRESENTATIVE LOGICAL ARCHITECTURE FUNCTIONS

ACCESS CONTROL
ADDRESS GENERATION
ADDRESS CONVERSION
LOCAL DISTRIBUTION, INCOMING
LOCAL DISTRIBUTION, OUTGOING
EXTERNAL NETWORK INTERFACE
SWITCHING
RED/BLACK CONVERSION
BLACK/RED CONVERSION
TRANSACTION PREPARATION ASSISTANCE

c. Development of Functional Architectures. The second phase of the methodology is the development of functional architectures. A functional architecture is a stratified logical architecture augmented by additional functions. The stratification consists of a hierarchical grouping of the functions in a logical architecture. After the stratification is accomplished, the set of functions represented in the logical architecture is augmented by an additional set of functions. These additional functions were not represented in the logical architecture because they are not in-line functions or because they are implementation dependent in the sense that their placement in the sequence of functions is dependent on the stratification. A representative sample of functions that would be introduced at the functional architecture level is presented in Table II.

TABLE II. REPRESENTATIVE FUNCTIONAL ARCHITECTURE FUNCTIONS

TRANSACTION STORAGE AND RETRIEVAL
TRANSACTION READDRESSAL
TRANSACTION EDITING
JOURNALING
DATA STORAGE AND RETRIEVAL
SPEED CONVERSION
CODE CONVERSION
FORMAT CONVERSION
USER IDENTIFICATION AND VERIFICATION

A hierarchy is a ranking, ordering, or grouping of entities (in this case functions) into a sequence of levels. In the case of a communications network shared by a large number of users, the following three hierarchical levels can be defined:

- Local Level - The local level is the level that contains the individual users and user terminals.
- Regional Level - The regional level would include elements that would interface with elements at the local level but would in general serve a subset of the total user community.
- Global Level - The global level consists of elements that serve the whole network.

The stratification of a logical architecture indicates at which level the functions are to be performed (without changing the order in which they are performed). This stratification enables the augmentation of the set of logical architecture functions by additional functions such as those in Table II.

d. Development of System Architectures. The third phase in the methodology is the development of system architectures. A system architecture is defined as the architecture that results from the allocation of functions within the hierarchical levels of a functional architecture to specific system elements. A system element is a conceptual entity, either hardware or software, that is responsible for some subset of functions.* A system architecture thus consists of system elements, a hierarchy into which they are configured (this hierarchy is derived from the functional architecture), and the relationships these elements have with each other.

* In defining system architectures, system elements are defined only in terms of their functions and not in terms of the specific technologies that implement these functions. For example, a system element might be defined as a terminal that is capable of performing certain functions but a specific make or model of terminal would not be indicated.

There are certain functions that were not appropriate to consider at the logical architecture level or the functional architecture level but need to be integrated into a system architecture. These functions include all system management and control as well as functions that depend on system level considerations. Table III is a representative list of such functions. The integration of these new functions requires consideration of the allocation of the functions that were embodied in the functional architecture to system elements.

TABLE III. REPRESENTATIVE SYSTEM ARCHITECTURE FUNCTIONS

CONCENTRATION
MULTIPLEXING
SECURITY PROCESSING
KEY VARIABLE DISTRIBUTION
NETWORK CONTROL
TRAFFIC CONTROL AND ROUTING
STATUS MONITORING
NETWORK MANAGEMENT
JOURNALING
DATA BASE MAINTENANCE
MESSAGE SWITCHING
PACKET SWITCHING
CIRCUIT SWITCHING
ENCRYPTION/DECRYPTION
ERROR CONTROL

e. Development of Network Architectures. The fourth phase of the methodology is the development of network architectures. A network architecture is defined as a system architecture together with specifications of connectivity policy among system elements, geographic distribution of system elements, and transmission media. A network architecture thus includes topological considerations indicating which elements are interconnected; numbers of elements, channels, and trunks; traffic parameters such as numbers of users and throughput; and performance parameters pertaining to reliability, survivability, and availability. A network architecture thus defines the network in sufficient detail to permit analysis of cost, benefits, and risks.

3. TECHNICAL APPROACH FOR PHASE 1

This section describes a methodology for synthesis and ranking of logical architectures. The starting point for this methodology is a given set of functions that need to be performed in order to accomplish the transfer of information among two or more persons or places. Table I presented a representative example of such a set of functions.

The set of logical architectures is a subset of the set of all finite sequences of functions. Note that an arbitrary finite sequence of functions may have the functions listed in any order and have some

functions appearing arbitrarily often. The relationship between a sequence of functions and a communications network is that each time information is transferred using the network, the functions are performed in the order in which they appear in the sequence.

The subset of the total set of sequences that are called logical architectures is the subset of sequences that are in some sense plausible. The notion of plausibility arises from the fact that in many cases it does not make sense to perform the functions in certain orders. In addition, it may not make sense for some functions to be the first or last functions in a sequence or for some functions to appear more or less than a certain number of times in a sequence. Examples illustrating the notion of plausibility were indicated in Section 2.2. The notion of plausibility thus embodies a set of constraints and a finite sequence of functions that satisfies all of these constraints is a logical architecture.

The methodology for synthesizing logical architectures starts by expressing the set of constraints as a regular grammar. A regular grammar is a set of rules such that all finite sequences constructed in accordance with these rules satisfy the constraints.* The formal definition of a regular grammar is presented in Appendix A. The advantage of this approach is that a regular grammar can be efficiently implemented as a computer program. This program, in turn, provides an efficient means of examining the large number of alternatives.

The next step in the synthesis and ranking of logical architectures is to implement the grammar as a computer program. This computer program produces sequences of functions that constitute logical architectures. As a consequence of this step, logical architectures can be synthesized. Appendix B describes a computer program labeled SYNTHESIS, that accomplishes this step.

---

* The notion of a regular grammar is motivated by the notion of a generative grammar for spoken language. A sentence in a spoken language is, by definition, a sentence constructed according to the rules of its generative grammar. Thus the string of words

"Bill John hit the in mouth"

is not a sentence because it is not constructed according to the rules of grammar. However, the string of words

"John hit Bill in the mouth"

is a sentence. Just as a generative grammar for English constructs sequences of words that are sentences in the English language, the regular grammar constructs sequences of functions that are elements of the logical architecture.

The approach to ranking logical architectures is to introduce a set of preferences that indicate that some logical architectures are better than others. This set of preferences can be used to indicate the relative worth of alternative logical architectures at the level of detail of a logical architecture. Note that the preferences can indicate only the relative worth of sequences of functions and cannot address questions pertaining to such issues as cost, schedule, or risk. It is for this reason that the synthesis and ranking of logical architectures is only the first phase of a four-phase process intended to produce a set of candidate architectures, that are then subjected to a complete evaluation.

The set of preferences is obtained by polling a group of knowledgeable reviewers. These reviewers interact with the computer program by receiving from the program a list of logical architectures* and ranking the logical architectures according to their knowledgeable perceptions of relative worth. The factors that induce these preferences are then deduced from the expert rankings and aggregated into a set of weights.

This set of weights can then be imposed on the grammar to produce a new grammar. This new grammar, called a weighted grammar, outputs logical architectures which are better than those produced by the old grammar in that they reflect the results of the rankings by the knowledgeable reviewers. This process of ranking could be repeated to refine the weights until the process converges to a set that truly reflects the aggregated opinion of the reviewers. Finally, a dynamic programming algorithm is used to determine the optimum logical architectures.

Note that the optimum logical architecture may not be unique and that a given optimum logical architecture may have more than one corresponding network architecture. Thus, in general, the four-step process described in Section 2 will yield a list of candidate network architectures which then must be evaluated with respect to all appropriate aspects of performance, cost, and risk. However, as a consequence of carrying out the synthesis and ranking procedure presented in this section, the candidate network architectures all correspond to optimum logical architectures. Furthermore, all possible logical architectures are at least implicitly considered in the ranking process that selected the candidates.

---

* These knowledgeable reviewers do not, in general, see a complete list of all logical architectures since this list is generally very large. What the reviewers see is a sample from this list.

The synthesis and ranking procedure described above is illustrated in Figure 1. In Appendix B, a set of three computer programs that implement this procedure are briefly described.

4. ILLUSTRATIVE EXAMPLE

This section presents an illustrative example of the methods presented in Section 3. This example indicates how to proceed from a set of functions and constraints to logical architectures. The mathematical techniques involved in this process are indicated in this section, but the mathematical calculations carried out for this example are not presented.

The process of determining optimum logical architectures given functions and constraints is broken down into the following three steps:

- Step 1. Infer a regular grammar that embodies the set of constraints
- Step 2. Determine a set of preferences or weights that measure the relative worth of logical architectures
- Step 3. Apply dynamic programming to obtain the optimum logical architecture.

For the purpose of this example, it is assumed that there are ten logical architecture functions. These may be functions such as those indicated in Table I, Section 2. However, the functions are simply denoted $f_1,\ldots,f_{10}$. The following constraints are imposed on these functions:

- Constraint 1 - Each function is to appear exactly once in a logical architecture. A consequence of this constraint is that the set of logical architectures is a subset of the set of permutations of the ten functions.
- Constraint 2 - The allowable positions of each function (i.e., the positions that each function may occupy in a permutation if the permutation is to be a logical architecture) are restricted to those indicated in Table IV. Thus for example, function f_1 may occupy any of the first four positions in a logical architecture and any permutation of $f_1,\ldots f_{10}$ that has f_1 in position five through 10 is not a logical architecture.

The problem is to infer from these constraints a regular grammar. A grammar is a set of rules for constructing sentences in a language. In this case, the sentences are logical architectures and the language is the set of all logical architectures. Since any useful language

The synthesis and ranking procedure described above is illustrated in Figure 1. In Appendix B, a set of three computer programs that implement this procedure are briefly described.

4. ILLUSTRATIVE EXAMPLE

This section presents an illustrative example of the methods presented in Section 3. This example indicates how to proceed from a set of functions and constraints to logical architectures. The mathematical techniques involved in this process are indicated in this section, but the mathematical calculations carried out for this example are not presented.

The process of determining optimum logical architectures given functions and constraints is broken down into the following three steps:

- Step 1. Infer a regular grammar that embodies the set of constraints
- Step 2. Determine a set of preferences or weights that measure the relative worth of logical architectures
- Step 3. Apply dynamic programming to obtain the optimum logical architecture.

For the purpose of this example, it is assumed that there are ten logical architecture functions. These may be functions such as those indicated in Table I, Section 2. However, the functions are simply denoted $f_1,\ldots,f_{10}$. The following constraints are imposed on these functions:

- Constraint 1 - Each function is to appear exactly once in a logical architecture. A consequence of this constraint is that the set of logical architectures is a subset of the set of permutations of the ten functions.
- Constraint 2 - The allowable positions of each function (i.e., the positions that each function may occupy in a permutation if the permutation is to be a logical architecture) are restricted to those indicated in Table IV. Thus for example, function f_1 may occupy any of the first four positions in a logical architecture and any permutation of $f_1,\ldots f_{10}$ that has f_1 in position five through 10 is not a logical architecture.

The problem is to infer from these constraints a regular grammar. A grammar is a set of rules for constructing sentences in a language. In this case, the sentences are logical architectures and the language is the set of all logical architectures. Since any useful language

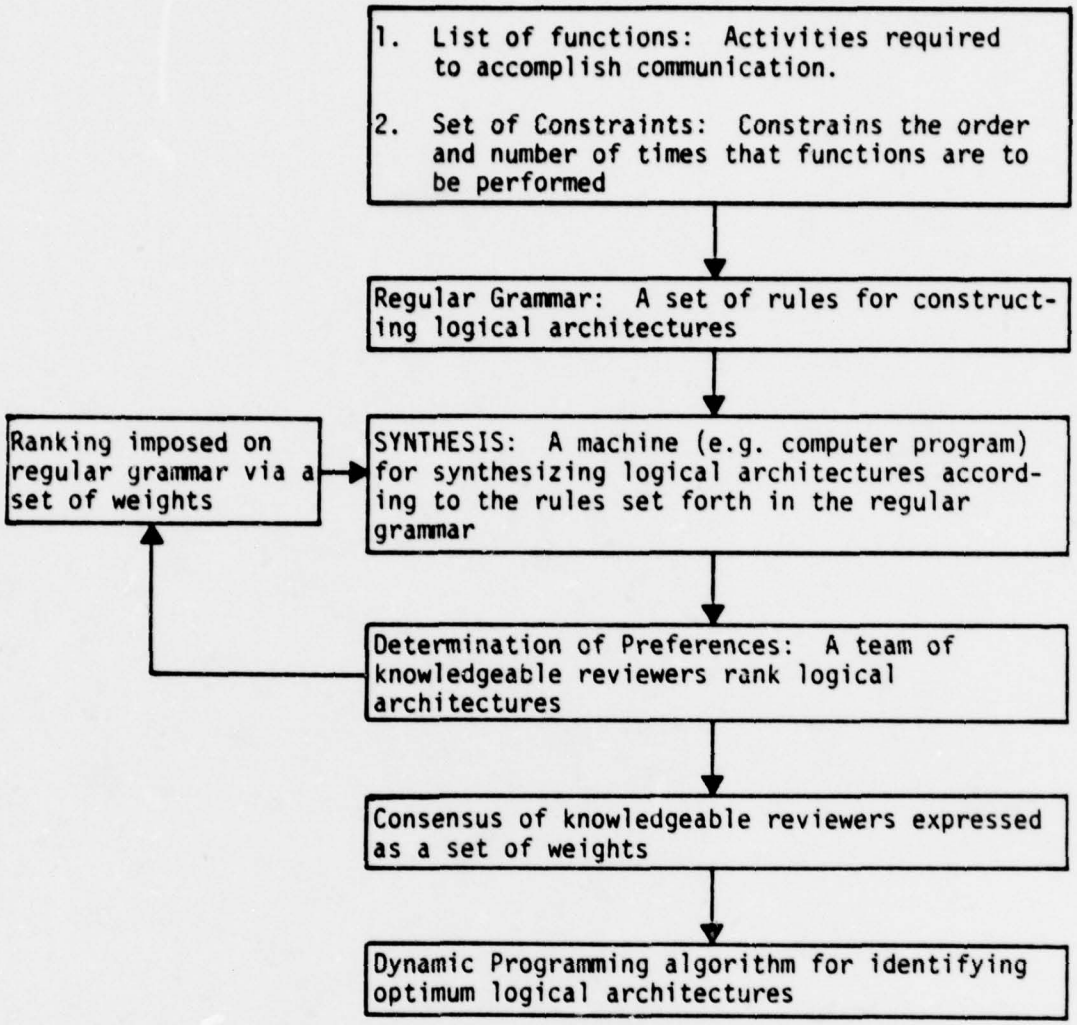


Figure 1. Synthesis and Ranking of Logical Architectures

contains many sentences, there are choices at each stage of the process of constructing a sentence. A grammar is an explicit set of rules that indicate precisely what the choices are at each stage.

TABLE IV. ALLOWABLE POSITIONS OF FUNCTIONS

FUNCTION	ALLOWABLE POSITIONS
f_1	1,2,3,4
f_1^1	1,2,3,4
f_2^2	2,3,4,5,6,7
f_3^3	7,8,9,10
f_4^4	4,5,6,7,8
f_5^5	4,5,6,7,8,9,10
f_6^6	4,5,6,7,8,9
f_7^7	2,3,4,5,6,7,8
f_8^8	7,8,9,10
f_{10}^9	1,2,3,4,5,6

An example of a rule that could be part of a grammar that embodies the above constraints is the following: start the string with f_1, f_2, or f_{10}. This rule indicates that the choices for the first position are the three functions whose allowable positions include the first position. A regular grammar is a grammar in which the rules have a special form. The advantage of a regular grammar is that the special form enables the grammars to be constructed and implemented by computationally efficient computer programs to generate the sentences in the language. The formal definition of regular grammar is presented in Appendix A.

After the regular grammar has been inferred from the constraints, the next step is to develop a set of weights that measure the relative worth of logical architectures. In the methodology the weights constitute probability distributions on the choices that are available at each step of the construction of a logical architecture. In this example if the first R steps of the construction of a logical architecture resulted in the string, fi_1, $fi_2 \ldots fi_R$, then the choices for the function fi_{R+1} are a subset (usually a proper subset) of the set of functions whose allowable positions include R+1. If this set of choices is [fji,..., f_{jm}] then the weights would include a probability distribution $[P_{j1},\ldots,P_{jm}]$ where:

$$\sum_{i=1}^{m} P_{ji} = 1$$

The reason that weights are formulated in this manner is that the implementation of the methodology includes a computer program (c.f. Appendix B) that randomly generates logical architectures. This means that in making the choices at each stage of the construction of a logical architecture, the program makes a random choice from the choice set (the choice set is $[f_{j1},\ldots,f_{jm}]$ in the example above). The weights become the probabilities of making each choice. An ideal set of weights would confer on the program a tendency to generate better logical architectures than another set of weights.

In order to approximate an ideal set of weights, an initial set of logical architectures is generated and shown to a group of knowledgeable reviewers, who rank them. New weights are inferred from the rankings. This process is repeated until a generally agreed upon set of weights is obtained. This final set of weights is imput to a dynamic programming algorithm that infers an optimum logical architecture.

REFERENCES

1. Martin, James. Telecommunications and the Computer, 2nd Edition, Prentice-Hall, Englewood Cliffs, N.J., 1976

2. Joint Chiefs of Staff (JCS) Memorandum of Policy (MOP) 165, AUTODIN and Associated Message Processing Systems, May, 1976

3. General Accounting Office (GAO), "Need to Consolidate Responsibility for Automatic Digital Network (AUTODIN) Terminals," B-169857, July 17, 1974.

4. Defense Communications Agency, "Integrated AUTODIN System Architecture", December, 1977

5. Booz, Allen Communications and Information Technology Division, "Integrated AUTODIN System Mid-Term Architecture Definition," Draft Report, December, 1978

6. Booz, Allen Communications and Information Technology Division, "Structure of a Methodology for Generation of IAS Candidate Architectures," DDC AD A057696

APPENDIX A

MATHEMATICAL CONCEPTS

APPENDIX A
MATHEMATICAL CONCEPTS

The purpose of this appendix is to present a formal introduction to the notions of regular grammars. This appendix is not a comprehensive presentation of this subject; rather it is a presentation of the definitions and basic results that underlie the methodology in the report. References 1 and 2 provide more complete discussions of this material.

If X is an arbitrary set, then a non-empty string in X is an element of the free semi-group generated by X. The empty set $\emptyset$ is regarded as a string called the empty string. The set of all strings (empty and non-empty) is a free monoid. The semi-group of all non-empty strings is denoted by X^+ and the monoid of all strings is denoted by X^*.

In order to define the concept of a regular grammar, the following notation is introduced:

V_T = a finite set of symbols called words
V_N = a finite set of symbols called syntactic variables
1 = a distinguished symbol in V_N called the start symbol
$V = V_T \cup V_N$
R = a finite subset of $V_N \times V^*$. R is called the set of rewrite rules.

Definition 1. If $u \in V^+$ and $v \in V^*$ then u is said to directly derive v if there are strings $i,j,x,y \in V^*$ such that $u = xiy$ and $v = xjy$ and $(i,j) \in R$.

Definition 2. If $u \in V^+$ and $v \in V^*$ then u is said to derive v if there is a finite sequence $Z_0, Z_1, \ldots Z_m$ V^* such that $u = Z_0$, $v = Z_m$, and Z_i directly derives Z_{i+1}.

Definition 3. A language is any subset of V_T^*

Definition 4. A phrase structure language is any language having the following property:

> There exists a set V_N and a finite set of rewrite rules, R, such that $u \in V_T^*$ is an element of the language if and only if 1 derives u.

A phrase structure language consists of all strings in V_T^* that are obtained by starting with 1 and successively making substitutions that are allowed by the set R of rewrite rules until the string contains no syntactic variables. The cardinality of the language is finite or countably infinite. Since the elements can be systematically enumerated, the language is also called recursively enumerable. The elements of a language are called sentences by analogy to the sentences in a spoken language.

Definition 5. A phrase structure grammar, or simply a grammar consists of V_T, V_N, and R as defined previously. The resulting language is called the language generated by G and denoted L(G).

Definition 6. A grammar G is said to be a regular grammar if all rewrite rules can be written as one of the following two types:

a. Continuing rules: a continuing rule has the form (i,xj) where $i,j \in V_N$ and $x \in V_T^*$

b. Terminating rules: a terminating rule has the form (i,x) where $i \in V_N$ and $x \in V_T^*$.

The terms continuing and terminating are used to indicate that the application of a continuing rewrite rule does not result in an element of L(G) whereas the application of a terminating rewrite rule does

result in an element of L(G). In Generating elements of L(G) by starting with the start symbol 1 and deriving a sequence $1 = Z_0, Z_1, \ldots, Z_m$ L(G) where Z_i directly derives Z_{i+1}, each of the strings Z_i has exactly one syntactic variable for $i < m$. The string Z_m has no syntactic variables and is a member of L(G) (i.e., it is a sentence).

APPENDIX B

IMPLEMENTATION APPROACH

APPENDIX B
IMPLEMENTATION APPROACH

The purpose of this appendix is to briefly describe three computer programs used to implement the methodology in this report. These programs are:

- SYNTHESIS - generates logical architectures given a set of constraints and an initial weight table
- CALIBRATION - generates a weight matrix, i.e., a set of weights given a ranking of a sample set of logical architectures by knowledgeable reviewers.
- SELECTION - chooses an optimal architecture using a dynamic programming algorithm.

Figure B-1 indicates how the three programs work together to produce an optimum logical architecture. SYNTHESIS accepts a set of constraints as inputs and, the first time it is run, uses a default preference matrix in which all possibilities are equally weighted at each stage of the process of synthesizing a logical architecture. The output of this first run is a sample set of logical architectures that are then shown to a group of knowledgeable reviewers who rank them. These rankings are translated into a new preference matrix (i.e., a new set of weights) by CALIBRATION. The new weight matrix is input back into SYNTHESIS to generate a new sample set of logical architectures and further refine the preference matrix. This process continues until an agreed upon preference matrix is obtained. This preference matrix is input to SELECTION, which then uses dynamic programming to obtain an optimum logical architecture.

SYNTHESIS accepts as inputs the set of words, the constraints, and a set of weights. If no weights are explicitly input (i.e., the first time SYNTHESIS is used in the process described above), then a default set of weights are used. The default weights give each choice of a word (given a state) equal weight. SYNTHESIS consists of the following four subprograms:

- ACCEPT - tests any string of functions against the constraints to determine if the string constitutes a logical architecture
- REWRITE - produces logical architectures by randomly generating strings of functions consistent with the constraints
- INFER - uses the logical architectures generated by REWRITE to build a regular grammar by an abduction process

. AUTOMATION - uses the regular grammar produced by INFER to generate logical architectures.

CALIBRATION accepts the ranking that a set of reviewers gave to a sample set of logical architectures. These rankings are then translated into a set of weights. The weights are passed to SYNTHESIS.

SELECTION selects an optimum logical architecture.

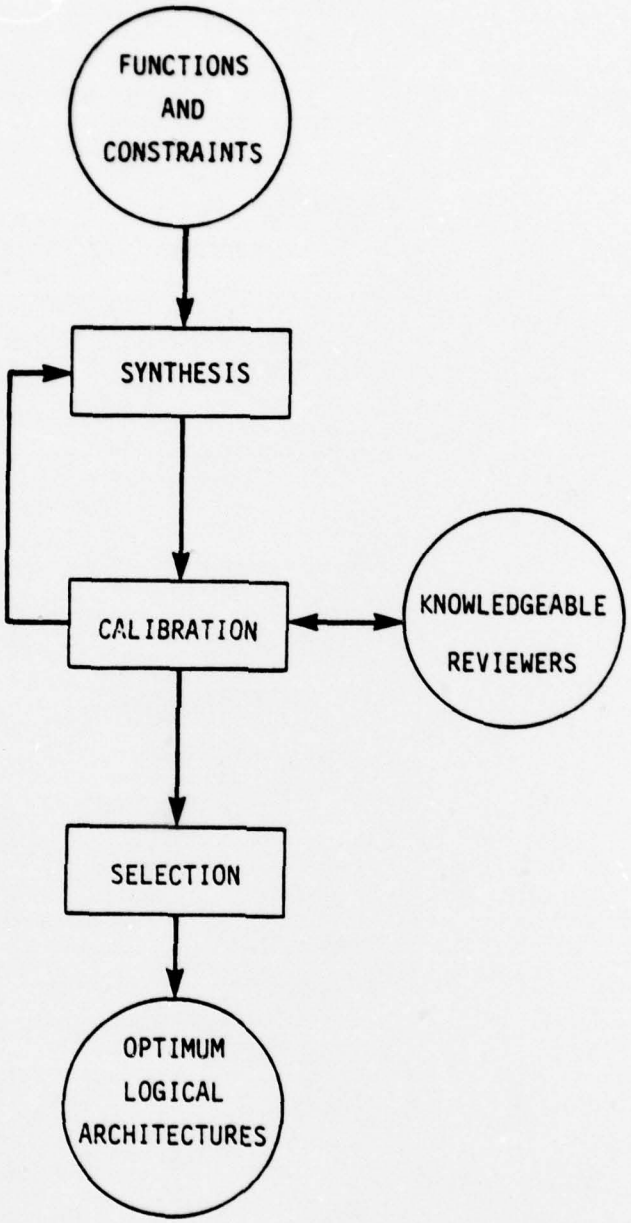


Figure B-1. Implementation Approach

APPENDIX C

DEFINITIONS OF IAS FUNCTIONS

APPENDIX C
DEFINITIONS OF IAS FUNCTIONS

This appendix presents definitions of various IAS functions. The functions are categorized as in-line, auxiliary, and management and control functions. These definitions are presented in Tables C-I, C-II, and C-III.

TABLE C-I. REPRESENTATIVE IN-LINE IAS FUNCTIONS

	FUNCTION	DEFINITION
1.	Address Generation	Address generation is the process of determining and assigning an address to a transaction to identify the ultimate destination of the transaction. It is normally performed by the originator of the transaction, but can also be performed by the network based on message content, predetermined addressing or other parameters.
2.	Address Conversion	Address conversion is the translation of addresses in one form, such as plain language, to another form, such as routing indicators or logical addresses, which are used by the network for routing transactions.
3.	Access Control	Access control refers to the procedures used by the network to grant or deny user access either to the network or to another user for data transmission.
4.	Local Distribution	Local distribution is the delivery of transactions directly to users which share a local geographic area or access element (as opposed to delivery of transactions through the network).
5.	External Network Interface	External network interfaces include the physical and functional interface elements, such as protocol and format translation, to allow the exchange of information between two separate networks.
6.	Switching	Switching is the process performed by a network element to select the appropriate output line or channel to transmit a message.
7.	Red/Black Conversion	Red/Black conversion is the translation of information from a non-secure form to a secure form. This conversion is usually accomplished by encryption.

TABLE C-I. REPRESENTATIVE IN-LINE IAS FUNCTIONS (Continued)

	FUNCTION	DEFINITION
8.	Black/Red Conversion	Black/Red conversion is the translation of information from a secure form to an unsecure form (e.g., decryption).
9.	Transaction Preparation Assistance	Transaction preparation assistance refers to the services performed by a network to aid the user in preparation of a transaction. Examples of these services include editing, formatting, and provision of message masks.
10.	Concentration	Concentration is the collection of information from a number of sources, delivery of the information to a common output channel, and the inverse of these two processes. A concentrator may employ a form of multiplexing. Concentrators typically perform more processing functions than multiplexers.
11.	Multiplexing	Multiplexing is the processing of combining data received over a number of channels onto one common output channel via frequency division or time division techniques. The inverse process, demultiplexing separates and distributes the data from the common channel to individual channels.
12.	Speed Conversion	Speed conversion refers to the process of receiving information at one speed and transmitting it at another speed (e.g., a switch receiving data at a low speed from a terminal and transmitting it at a high speed to another terminal.
13.	Code Conversion	Code conversion is the translation of data from one code to another to allow users operating in different codes to communicate.
14.	Format Conversion	Format conversion refers to the process of translating the format of a transaction to one which is acceptable to the recipient, to allow users which process different formats to communicate.

TABLE C-I. REPRESENTATIVE IN-LINE IAS FUNCTIONS (Continued)

FUNCTION	DEFINITION
15. Mode Conversion	Mode conversion refers to the process of receiving information on a channel employing one mode or protocol and transmitting it over another channel employing another mode or protocol.

TABLE C-II. REPRESENTATIVE AUXILIARY IAS FUNCTIONS

	FUNCTION	DEFINITION
1.	Transaction Storage and Retrieval	Transaction storage and retrieval refers to the service provided by a network element which stores complete transactions, for accountability and reference purposes, and makes the transaction available for subsequent retrieval. Storage may be on-line or off-line and is often maintained for long periods of time, e.g., 30 days. Retrieval may be accomplished by local operator actions or remote user request.
2.	Transaction Readressal	The transaction readressal function allows messages held in storage to be readdressed for additional distribution upon direction of a local operator or remote user.
3.	Transaction Editing and Reformatting	Transaction editing and reformatting includes all the functions that a network element might provide to allow the form or content of messages to be changed. These functions may be accomplished automatically or manually.
4.	Data Storage and Retrieval	Data storage and retrieval refers to the service provided by a network element which makes files of data available to users for manipulation and retrieval (as opposed to transaction storage and retrieval which stores user-to-user traffic).
5.	User Identification and Verification	User identification and verification is the process of validating the identification of a subscriber or authorization for the subscribers to use the system or to gain access to certain files, elements, or processes (e.g., via codewords, identification numbers or personal characteristics).
6.	Multiple Delivery	Multiple delivery is the delivery of individual transactions to multiple users. Delivery determination may be based on addresses or other elements of the transaction.

TABLE C-III. REPRESENTATIVE MANAGEMENT AND CONTROL IAS FUNCTIONS

	FUNCTION	DEFINITION
1.	Network Control and Management	Network control and management refers to the collection of functions required to control the network configuration and insure network availability to users. It includes such functions as addition and deletion of circuits, performance assessment and on-line diagnostics.
2.	Journaling	Journaling is the process of recording, for accountability purposes, information to identify, trace and retrieve transactions. Journaling may include storage of the complete transactions.
3.	Traffic Control and Routing	Traffic control and routing functions are those functions necessary to control the flow of traffic in the network. Included are such functions as precedence processing, connection control, alternate routing and routing table updates.
4.	Status Monitoring	Status monitoring includes monitoring of network and element operation, collection of statistics and reporting for the purposes of fault detection, restoral and system management.
5.	System Security	System security refers to all the functions necessary to allow users to exchange classified information and to protect the information in the network.

REFERENCES

1. John E. Hopcraft and Jeffrey D. Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, Reading, 1969

2. Stefan Shrier, "Abduction Algorithms for Grammar Discovery." Ph.D. Dissertation, Brown University, 1977.